2005－CVIM－149（29）
2005／5／12

PC クラスタを用いた実時間３次元形状復元のための並列パイプライン視体積交差法

ウ　小軍[†]　松山隆司[††]

コンピュータビジョンの分野において, 人物動作のモニタリングは基本的なタスクの一つである. 動作解析, マンマシンインタラクション, あるいは 3D ビデオなどといったさまざまな場面で, 2 次元の観測画像のみならず, 対象の詳細な 3 次元形状情報が望まれる. 特に, ダイナミックなマンマシンインタラクションや仮想空間での臨場感高い共同作業支援などには, 実時間 3 次元形状復元は不可欠である. 本論文では, 形状復元の計算量を削減するため, 視体積交差法に基づき, 高速かつ安定に計算可能な 3 基準平面視体積交差アルゴリズムを提案する. 同時に, 当アルゴリズムを並列化し, PC クラスタを用いた並列計算による速度向上が図られた. さらに, 計算効率をあげるため, 並列パイプライン処理モデルを導入した. 並列パイプラインの実装に関して, スレッドツリーモデルを提案し, それによって, 計算量に応じたスレッドスケジューリングが容易になり, ダイナミックに変化する対象姿勢による形状復元の計算量変動にも安定したスループットが維持できるようになる. 評価実験によって, ボクセルサイズが 5mm ×5mm ×5mm の場合は, 約 6 fps, 10mm × 10mm × 10mm の場合は 20 fps 以上のパフォーマンスが確かめられた.

Parallel Pipeline Volume Intersection for Real-Time 3D Shape Reconstruction on a PC Cluster

Xiaojun Wu [†] and Takashi Matsuyama[††]

The human activity monitoring is one of the major tasks in the field of Computer Vision. Not only the images in appearance, but also the fine 3D shape of the target are desired for studies like motion analysis, man-machine interaction technologies, the 3D video technologies and so on. Especially, to realize multi-modal and dynamic interactions between human-beings and machines, or to realize the remote collaboration in the virtual reality, it is desired to acquire the 3D shape in real time. In this paper, we focus on the real-time 3D shape reconstruction of a moving person by accelerating and parallelizing the volume intersection method on a PC cluster. Ideas are proposed in aspects of both the fast algorithm and the implementation model to realize the efficient parallel pipeline on a PC cluster. While the plane based volume intersection accelerates the shape computation efficiently, the efficiency will be damaged by certain camera layouts, where the viewing direction is nearly parallel to the base plane. To avoid such break down, we extend the plane based volume intersection to the 3-base-plane volume intersection so as that the computation can be carried out stably and efficiently for arbitrary camera layouts. The parallelization of the 3-base-plane volume intersection is also conducted for the parallel computation on a PC cluster. On each PC the pipeline processing model is introduced to achieve the maximum throughput. Since the 3D shape of the target changes with the motion, the computational complexity changes in real time. To keep high throughput while the computational complexity changing, we propose the tread tree control model for the implementation of the parallel pipeline, where each pipeline stage is implemented as one thread and all threads are organized as a thread tree. The dynamic scheduling of the threads is realized by changing the structure of the thread tree. By dynamically scheduling the threads according to the estimated computational complexity, the throughput of the pipeline can be achieved as high as possible. From the performance evaluation results, for the voxel size of 5mm × 5mm × 5mm, the throughput of about 6 volumes per second is achieved on a PC cluster of 27 dual Pentium III 1Ghz powered PCs, and for the voxel size of 10mm × 10mm × 10mm, the rate of over 20 volumes per second is achieved.

1. Introduction

The human activity monitoring is one of the ma-

[†] NTT サイバースペース研究所
NTT Cyber Space Laboratories
[††] 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University

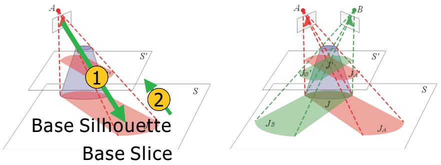

Fig. 1 Plane-based volume intersection method

jor tasks in the field of Computer Vision. The detailed 3D shape of the human motion is required in real time for the motion analysis, the action understanding and so on. The techniques of the 3D shape modeling are also desired to realize the free viewpoint video or the 3D video contents[1)∼4),6),7),9)∼11)]. Furthermore, to realize dynamic interactions between human-beings and machines, or to realize the high reality remote collaboration in the virtual space[8)], the 3D shape of the target must be acquired in real time. Therefore, we focus on the real-time 3D shape reconstruction of a moving person.

The volume intersection method(VIM)[5)] is adopted as the basic algorithm for the *full* 3D shape reconstruction from multi-viewpoint silhouettes. For the real-time computation, we propose the plane based volume intersection[12)] to accelerate the VIM, where the 3D space is decomposed into parallel planes(slices) so that the shape are calculated as the cross-sections on each slice. The algorithm is summarized as following, and is illustrated in Figure 1.

(1) **Base Silhouette Generation(BSG)** : Project the object silhouette observed by each camera onto a common base plane (Figure 1 left, ①).

(2) **Visual Cone Generation(VCG)** : Project each base plane silhouette onto the other parallel planes (Figure 1 left, ②).

(3) **Visual Cone Intersection(VCI)** : Compute 2D intersection of all silhouettes projected on each plane (Figure 1 right).

While the plane based volume intersection is efficient at reducing the computational complexity of the volume intersection, it is obvious that the computation of the BSG will break down if any input screen of the multi-viewpoint cameras is nearly perpendicular to the base plane. That is, in case of the base plane being statically determined, the computation can not be carried out for arbitrary camera layouts.

In this paper, we propose the 3-base-plane volume intersection, which is an extension of the plane based volume intersection. By this extension, not only the computational breakdown due to the camera layout can be avoided, but also the upper bound of the computational complexity of the BSG can be estimated, which is important for the real-time computation.

We also show the parallel algorithm for the 3-base-plane volume intersection, and propose the parallel pipeline processing model on a PC cluster to get high throughput of the volume intersection. Since the 3D shape of the target changes with the motion, the computational complexity changes in real time. To keep high throughput while the computational complexity changing, we propose the tread tree control model for the implementation of the parallel pipeline, where each pipeline stage is implemented as one thread and all threads are organized as a thread tree. The dynamic scheduling of the threads is realized by changing the structure of the tree. By dynamically scheduling the threads according to the estimated computational complexity at each time, the throughput of the pipeline can be achieved as high as possible.

In what follows, we first describe the 3-base-plane volume intersection and its parallelization in details. After that, the thread tree control model is shown. Experiments of tuning the executing order of the threads by changing the structure of the thread tree are also conducted. At last, the performance evaluation experiments are shown to prove the efficiency of the proposed algorithm and the implementation model.

2. The 3-Base-Plane Volume Intersection

As mentioned before, the plane based volume intersection method can exhibit a serious problem: when the optical axis of a camera is nearly parallel to the base plane, the size of the projected silhouette becomes very huge, which damages the computational efficiency. In the worst case, i.e. when the optical axis becomes parallel to the base plane, the projection cannot be computed and the method breaks down.

In the case of static camera arrangements, it is possible to select the base plane so that the worst case can be avoided. But the question of which base plane is optimal for computation remains open. In the case of dynamic camera arrangements, we have to extend the method to keep the computational

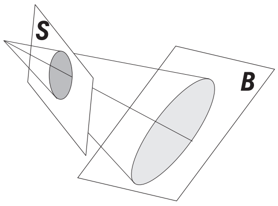

Fig. 2 Projection from an input image screen to the base plane

efficiency as well as avoid the worst case.

Here we first analyze how the computational cost changes depending on the base plane selection and then propose an augmented plane-based volume intersection method for flexible camera-work setup.

Suppose an observed object silhouette on an input image screen S is represented as a circle of radius r, which is projected onto the base plane B as an ellipse (Figure 2). Let θ, f, l denote the dihedral angle between B and S, focal length, and the distance from the projection center to B respectively. The area size s of the projected silhouette becomes:

$$s = \pi r^2 \cdot R(\theta, f, l), \tag{1}$$

$$where \;\; R(\theta, f, l) = \frac{l^2}{f^2 \cos\theta}.$$

Thus, the projected silhouette is expanded in the ratio of $R(\theta, f, l)$. It is clear that R is monotonically increasing with θ and diverges to infinity at $\theta = \pi/2$, which corresponds to the worst case.

In the case of a static arrangement of n cameras, let $\{f_1, f_2, \ldots, f_n\}$, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ and $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ denote the focal lengths, view directions and positions of the cameras respectively. The base plane can be represented by its normal vector $\mathbf{D} = (D_x, D_y, D_z)^t$ from the origin of the world coordinate system. Given the base plane, θ_i and l_i for each camera can be calculated from $\mathbf{v}_i$, $\mathbf{p}_i$ and $\mathbf{D}$. Let $\{a_1, a_2, \ldots, a_n\}$ denote the area sizes of object silhouettes observed by the cameras. From equation (1), the area size of each projected silhouette becomes:

$$s_i = a_i \cdot R_i = a_i \cdot \frac{l_i^2}{f_i^2 \cos\theta_i} \tag{2}$$

So the optimal selection of the base plane can be achieved by solving

$$\mathbf{D} = \mathrm{argmin} \sum_{i=1}^{n} s_i. \tag{3}$$

Note firstly that it is hard to solve this problem analytically, because we cannot represent $\{a_1, a_2, \ldots, a_n\}$ in analytical forms. Moreover, since $\{a_1, a_2, \ldots, a_n\}$ change dynamically depending on the object action, the optimal base plane selection should be done frame by frame, which introduces a large overhead. Note also that if we changed the base plane frame by frame, we would have to map each reconstructed 3D volume to a common coordinate system since the 3D volume is represented in such coordinate system that is defined by the base plane. This coordinate transformation would also introduce a large overhead.

In[4], to synthesize free-viewpoint video of the sports, the plane-based volume intersection is conducted. To avoid the above breakdown of the projection, the direction of the parallel plane changes dynamically keeping perpendicular to the viewing direction. Since in[4], no explicit 3D shape is desired, they do accelerate the projection by decreasing the plane number and changing the space resolution according to the viewpoint of the watcher. In our case, however, the full 3D shape is desired in real time, any non-linear coordinates conversion that increase the computation complexity should be avoided.

2.1 Augumented PPPP Algorithm — 3-Base-Plane Algroithm

To avoid the case that the determined base plane is parallel to the direction of the camera, we can also determine multiple candidate base planes for the multi-viewpoint camera system. For each camera, by selecting one from the candidates, the worst case can be avoided. Although any two non-parallel planes are enough to avoid the worst case theoretically, we determine the candidates as 3 mutually orthogonal planes, each of which are perpendicular to the axes of the world coordinates, for low overhead of the coordinates conversion. The augmented 3-base-plane volume intersection is shown as follows:

(1) **B**ase Plane Selection & Base Silhouette Generation(BSG): By the base plane selection, a set of cameras are partitioned into 3 groups: cameras in each group share the same base plane.

(2) **V**isual Cone Generation (VCG): Project the base object silhouettes onto each slice.

(3) **V**isual Cone Conversion (VCC): Convert the visual cone of each camera into the silhouettes on slices with the same direction.

(4) **V**isual Cone Intersection (VCI): All visual cones are intersected to generate the complete 3D object shape.

2.2 Computational Complexity of the Base Silhouette Generation in the 3-Base-Plane Volume Intersection

With the 3-base-plane volume intersection method, since $\mathbf{D}_x(t)\perp\mathbf{D}_y(t)\perp\mathbf{D}_z(t)$,

$$\min(\theta_{i_{\mathbf{D}_x(t)}},\theta_{i_{\mathbf{D}_y(t)}},\theta_{i_{\mathbf{D}_z(t)}})) \leq \sin^{-1}(\sqrt{3}/3). \quad (4)$$

This means that from equation (1), the area size of a projected object silhouette is bounded by $\frac{\sqrt{6}}{2}\cdot\frac{l_i(t)^2}{f_i^2}a_i(t)$. That is, by this extension, not only the worst case can be avoided, but also we can estimate the upper bound of the computational cost, which is very important for the design of the real-time active 3D shape reconstruction.

2.3 Evaluation of the 3-Base-Plane Volume Intersection

In this section, we evaluate the performance of the 3-base-plane volume intersection. 9 cameras of our multi-viewpoint video capturing system are used for the evaluation experiments. According to 3-base-plane volume intersection, the base plane of each camera is determined by each viewing direction, and is shown in Table 2. All the evaluations here are done in single thread programming. The specification of the PC is shown in Table 1. And the bounding box is fixed at 100 × 100 × 200[cm^3].

Table 1 PC Specification

CPU	Pentium III , 1GHz × 2
Memory	1GByte

Table 2 Base Plane of Each Camera

Camera ID	Determined Base Plane
Camera 1	ZX-plane
Camera 2	YZ-plane
Camera 3	XY-plane
Camera 4	ZX-plane
Camera 5	YZ-plane
Camera 6	XY-plane
Camera 7	ZX-plane
Camera 8	YZ-plane
Camera 9	ZX-plane

2.3.1 Direction of Base Plane

First, we select a single silhouette as the input silhouette to measure the base silhouette generation time while changing the angle between the base plane and the input screen. The voxel size is set as $r = 10$[mm]. Changing the angle between the base plane and the input screen from 0 to 86 [degree] in step 1 [degree], the time measured is plotted in Figure 3.

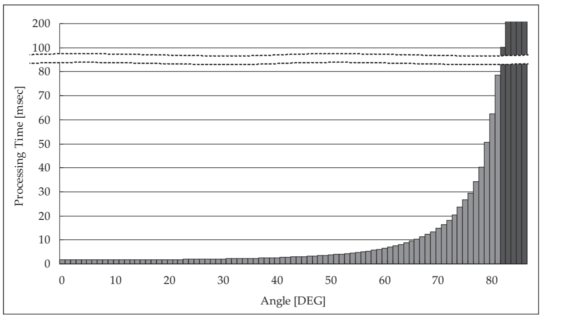

Fig. 3 Evaluation of Base Silhouette Generation (1)

The time graph shows that, while the PPPP algorithm performs well when the base plane is nearly parallel to the input screen, the projection breaks down when the base plane becomes nearly perpendicular to the input screen.

2.3.2 Evaluation of Base Silhouette Generation with Single-Base-Plane and 3-Base-Plane Methods

In this experiment, picking multi-viewpoint silhouettes as the input, the base silhouettes for all 9 cameras are generated by 2 different methods, i.e. the single-base-plane method where the base plane is fixed as the XY-plane, and the 3-base-plane method where the base plane is determined from the XY, YZ, ZX-planes. The details of the determined base plane for each camera is shown in Table 2. The evaluation result is plotted as bar graphs shown in Figure 4, where the base silhouette generation time of the single-base-plane method and the 3-base-plane method is plotted for each camera. From the graph, for almost cameras, the time of the 3-base-plane method is much shorter than that of the single-base-plane method. And as the worst case, i.e. the case of Camera 1, 2, 5 and 7, the PPPP breaks down, as labeled in the figures. The results prove that by the 3-base-plane method, not only the breakdown is avoided, but also the performance is improved.

In following experiments, the total processing time for one frame shape reconstruction is measured, with input silhouettes from the 9 cameras. The total processing time is obtained by adding all the time of each phase together, i.e the base silhouette generation (BSG), the visual cone generation (VCG), the visual cone conversion (VCC),and the visual cone intersection (VCI). The input silhouettes of 5 different postures are used in the exper-

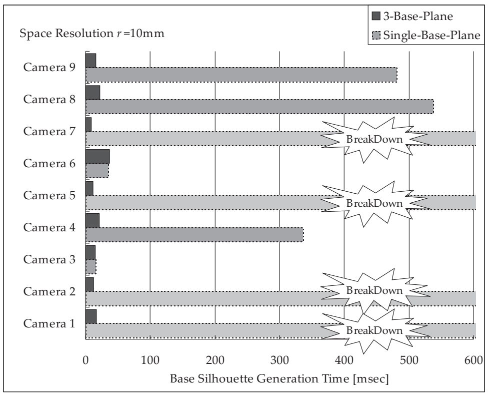

Fig. 4 Evaluation of Base Silhouette Generation, $r = 10$[mm]

iments and for each posture, the voxel size is set as $r = 5$[mm], $r = 10$[mm] and $r = 20$[mm]. Figure 5 shows the results as the stack bar graphs for 5 different postures. Higher the space resolution, longer the total processing time become. It is clear the visual cone generation rules the total processing. Meanwhile, the processing time changes while the posture changes.

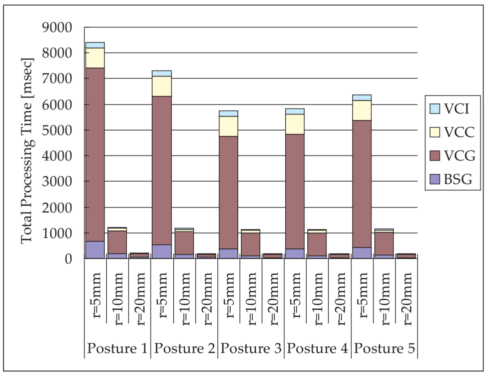

Fig. 5 Total Processing Time for One Frame Reconstruction

As the summary, we come to the following conclusions:

(1) The 3-base-plane augmented algorithm avoid the breakdown effectively.

(2) The visual cone generation is the heaviest method in the plane-based volume intersection.

(3) The performance are affected by the posture of the object, i.e. the computational cost changes dynamically in the real-time 3D shape reconstruction of a moving person.

(4) To improve the total throughput, it is effect to distribute the processing of the visual cone generation.

3. Parallelization of the 3-Base-Plane Volume Intersection

For the parallel processing, we apply the PC cluster architecture for the computation. The base silhouette duplication parallel algorithm is proposed as follows, and is shown in Figure 6:

(1) Image Capturing (IC)

(2) Input Object Silhouette Generation (ISG)

(3) Base Silhouette Generation (BSG)
On each camera node, generate the base silhouette on the selected base plane.

(4) Base Silhouette Duplication (BSD)
Suppose the cameras are grouped at least 2 groups, the base silhouette duplication is conducted as follows.

  (a) Grouping of Computation Nodes
  Assign computation nodes to the 3 camera groups to balance the computational cost for the visual hull generation of the 3 camera groups.

  (b) Duplicate the base silhouettes among the nodes in the same group.

(5) Visual Cone Generation and Intersection (VCGI)
Within each group, divide the slices into subsets by the node number of the group. On each node, generate the subsets of the visual cone and compute the intersections of them to get the subset of visual hull on each node.

(6) Visual Hull Conversion (VHC)
On each node, convert the subset of the calculated visual hull to the cross-sections perpendicular to one of the axes, for example, the Z-axis.

(7) Transfer Visual Hull (TVH)
Transfer the converted visual hull to the nodes in the camera group whose base plane is perpendicular to the Z-axis.

(8) Visual Hull Intersection (VHI)
On each node which received the visual hulls, compute the intersection of the subset of the visual hulls to obtain the subset of the 3D shape.

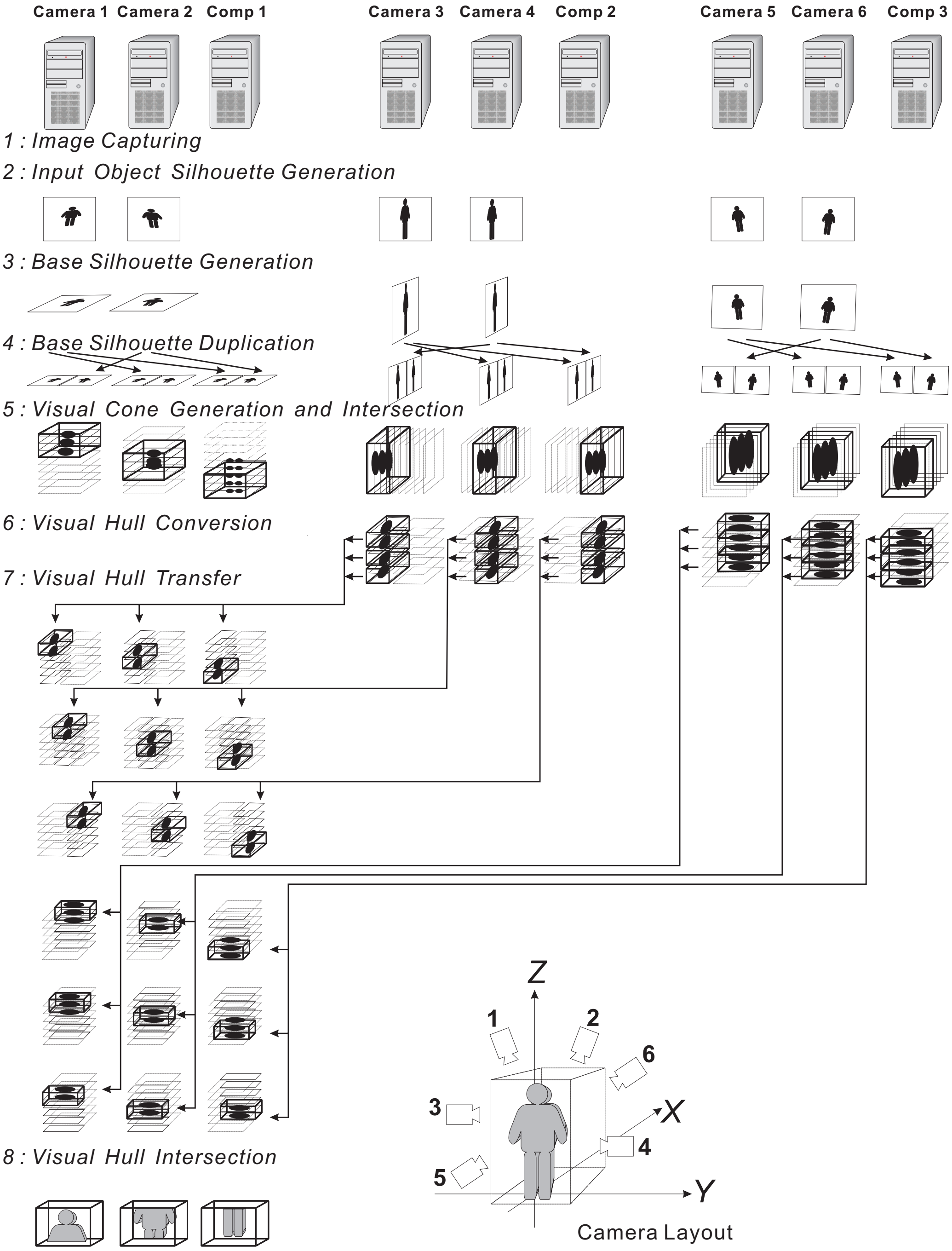
Camera 1
Camera 2
Comp 1
Camera 3
Camera 4
Comp 2
Camera 5
Camera 6
Comp 3
1 : Image Capturing
2 : Input Object Silhouette Generation
3 : Base Silhouette Generation
4 : Base Silhouette Duplication
5 : Visual Cone Generation and Intersection
6 : Visual Hull Conversion
7 : Visual Hull Transfer
8 : Visual Hull Intersection
Z
X
Y
1
2
3
4
5
6
Camera Layout

Fig. 6 Parallel Algorithm for The 3-Base-Plane Volume Intersection

4. Parallel Pipeline Implementation of the Plane Based Volume Intersection

4.1 Parallel Pipeline Processing Model

By the parallel algorithm shown above, the computation of the 3-base-plane volume intersection is decomposed onto each PC of the cluster. The processes assigned on each PC can then be categorized as follows, which is also summarized in Table 3.

- **Computation Phase**
  The computation phase is the step where the process time is mainly spent for arithmetic in-

Table 3 Process Type on Camera & Computation Node

Step	Camera Node			Computation Node		
	Computation	I/O		Computation	I/O	
		Local	Remote		Local	Remote
1		IC				
2	ISG					
3	BSG					
4			BSD			BSD
5	VCGI			VCGI		
6	VHC			VHC		
7			TVH			TVH
8	VHI			VHI		

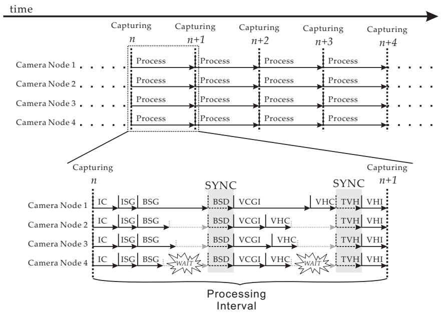

Fig. 7 Sequential Processing Model on Node PC

structions. For the camera node, the steps of ISG, BSG, VCGI, VHC and VHI are categorized as computation phases. For the computation node, computation phases consists of the steps of VCGI, VHC and VHI.

- **Data I/O**
  The data I/O phase is the step where the process time is mainly spent for reading or writing data between the memory and the I/O devices, for example, storages, the capture board, the network card and so on. While such processes need little CPU power on recent PC architecture, the computation phases is stopped by these processes. On a PC cluster system, the data I/O phases consists of 2 types as local I/O and remote I/O. In the above parallel algorithm, for the camera node, the local I/O phase contains the IC step and the remote I/O phases contain the BSD and TVH steps. For the computation node, all data I/O phases are the remote I/O phases which contain the BSD and TVH steps.

Notice that, while the time spent for the data I/O may be shortened by applying high speed devices, the remote I/O causes synchronization between nodes on a PC cluster, and it may cause extra wait time between the computation phases.

Figure 7 shows an example of time flow for 4 camera nodes executing the parallel 3-base-plane volume intersection. On the top of the figure, at each capturing time, the nodes are synchronized. And after the capturing, the computation is distributed on each node. We call the processing between 2 capturing on each node as the processing cycle, and the details of one processing cycle are shown at the bottom of the figure. On each node, the processes are carried out sequentially. At the start, all nodes are synchronized for the image capturing. Since the computation amount of BSG and VCGI depend on the size of the input silhouette and the camera layout, the time for these phases differs on each node. While the end timing of these phases on each node are different, the end timing of the BSD and the TVH are synchronized by the communication between nodes. It is clear that extra wait time is caused by the synchronization, and such wait time will not be shortened even though the time for BSD or TVH are shortened.

To get rid of such extra wait time, we propose the pipeline processing model on each node PC. That is, on the PC cluster, the parallel pipeline processing is carried out like Figure 8 shows.

In the figure, the processing flow of 4 camera nodes is shown on the top, and the details of the processing cycle on one node is shown at the bottom. On each node, the 8 steps of IC, ISG, BSG, BSD, VCGI, VHC, TVH and VHI are executed as 8 stages of the pipeline processing, i.e. each stage is the processing step for each of the continuous 8 frames respectively. In this parallel pipeline processing model, all nodes are only synchronized for the image capturing at the start of each cycle. After that, processes on each node are carried out concurrently. Since both the computation phases and the data I/O phases are executed simultaneously in one cycle, the computation phases will not

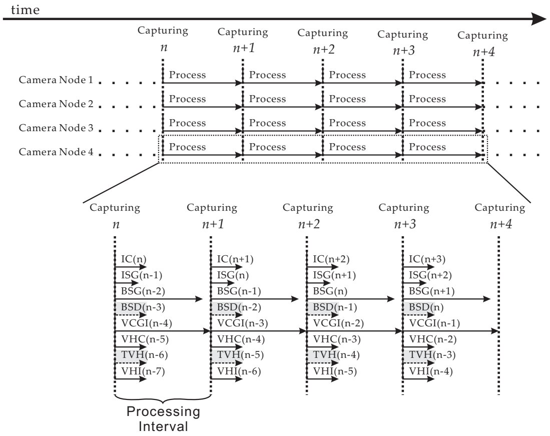

Fig. 8 Pipeline Processing Model on Node PC

be stopped by data I/O, including the remote I/O phases. That is, no extra wait time is spent during one cycle in the parallel pipeline processing model. Although the process for one frame is accomplished in multiple cycles, the processing interval is shortened from the summation of all steps + extra wait time to the time of the stage which runs longest.

To realize the pipeline processing model shown in Figure 8 on each node, the concurrent programming platform is required. While most recent operating systems support concurrent programming, there exist the following problems in practice:

- **Hardware Resource Conflict**
  As Figure 8 shows, the BSD and the TVH stages need access to the network devices. If only one network card is available for accessing, the two stages can not be executed at the same time. That is the mechanism for avoiding such resource conflict is required.
- **Limitation of CPU Resource**
  In most recent operating systems, the concurrent programming is realized by the time sharing scheduling (TSS) mechanism. That is, each full program is split into multiple executive units, and by switching the executive unit on CPU in a tiny time interval, multiple programs are executed concurrently in appearance. On such OS, the number of the units concurrently invoked at exactly same time will never be greater than the number of the CPU. Suppose each stage in Figure 8 is the executive unit, in practice, if the CPU number is less than 8, all 8 stages will never be executed at exactly same time, but be executed in some order determined by the OS scheduling mechanism.
- **Stage Order**
  Since not all stages can be invoked at exactly same time in practice, the invoking order of the stages will affect the process time of one cycle. If the time of each stage is static, the optimal order can be determined beforehand. But as mentioned so far, the process time changes dynamically while the shape of the object changes. So the mechanism to change the stage order dynamically is required.

As the summary, both the mechanisms of avoiding resource conflict and managing the stage order are required to realize the parallel pipeline processing for the real-time 3D shape reconstruction.

Furthermore, the parallel pipeline processing model is suitable for a general parallel computation on a PC cluster architecture because the extra wait time caused by the data sharing can be got rid of. Also the problems of the resource conflict and the stage order, i.e. the order of executive unit, are also general problems for the concurrent programming on most recent PCs.

In what follows, the system design to realize the parallel pipeline processing on a PC cluster is shown in details.

4.2 Multi-thread Implementation of Multiple Stages

To realize the parallel pipeline processing on the PC cluster, the following types of modules are designed.

- **P**rocessing Module
  This module consists of the memory addresses for the input and the output, and the program routines of one step in the parallel algorithm. The processing module can be categorized as the following 3 types : Computation Module, Local I/O Module and Communication Module. Each phase of the parallel algorithm is implemented as one processing module and is bound to one thread.
- **Control Module**
  To realize the parallel pipeline processing, not only the input and the output of each module should be assigned correctly, but also the mechanisms of avoiding resource conflict and managing stage order should be realized. While each processing module is bounded to one thread, the control module is designed to manage the threads and to realize the above tasks. The details of the control module will be described in the next section.

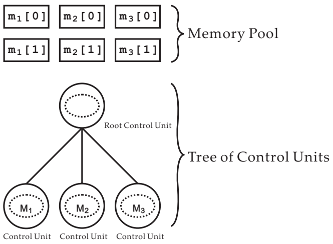

Fig. 9 Overview of The Control Module

4.3 Thread Tree Control Model for Threads Scheduling

The overview of the control module is shown in Figure 9. The control module consists of a memory pool and multiple control units which are organized in a tree structure. The details of these items are shown below.

By keeping double memory blocks for each phase, each phase can be executed concurrently. By alternating the buffer index between 0 and 1, the last output of the precede phase is given to the subsequent phase as the input, thus just realizes the pipeline processing. By applying this input & output assignment for each cycle, each processing module realizes one stage of the pipeline processing.

4.3.1 Control Unit

For convenience, in what follows, we suppose the multi-thread programming is supported on PCs. The control unit is just one thread and the main routine of the thread, i.e. the control routine, is defined below.

The control module consists of multiple control units and all these units are organized as a tree structure, which is realized by the data structure of the control unit as shown as below. Notice that the structure of the root control unit differs from others and is shown later.

The following are the details of the structure of the control unit.

- Parent Handle
  The handle of the control unit which is the parent of this control unit.
- Children Handle List
  The list contains the handles of the control units which are the children of this control unit.
- Processing Module Handle
  By this handle, one processing module is bound with the control unit. The computation or the communication routines of the module can then be called by the control unit through this handle.

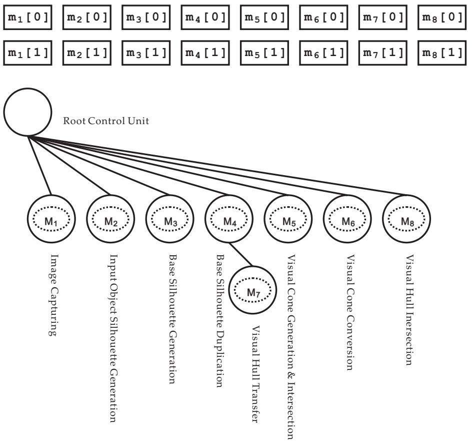

Fig. 10 The Control Module for The Parallel Pipeline of 3-Base-Plane Volume Intersection (Camera Node)

- Control Routine
  As mentioned above, the control routine is the main routine of the thread. When the thread is created, the following cycle is executed to realize the controlling task.
  (1) Sleep until being waken up.
  (2) Call the routine of the bound module, using the processing module handle.
  (3) When the routine of the bound module ends, wake up all of the child control units.
  (4) Sleep until all child control units finish one cycle.
  (5) Assign the input and the output for the bound module for the next cycle.
  (6) Inform the parent control unit the cycle finishing of this unit.

And the structure of the root control unit is shown below.

- Children Handle List
  This is the same as the common control unit.
- Cluster Synchronization Routine
  These routines are used to synchronize the pipeline processing with other nodes in the PC cluster.
- Control Routine
  When the program starts, the following cycle is executed in the root control unit.
  (1) Call the cluster synchronization routine.
  (2) Wake up all of the child control units.
  (3) Sleep until all child control units finish one cycle.

4.3.2 Summary

Based on the above thread tree model, the par-

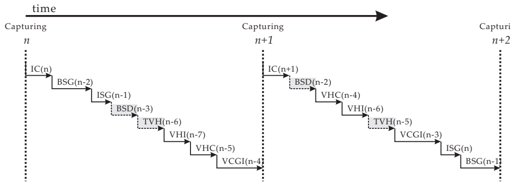

Fig. 11 Processing Flow of 3-Base-Plane Parallel Pipeline Processing

allel pipeline of the 3-base-plane volume intersection is implemented as the tree shown in Figure 10, which shows the diagram of the control module for the camera node. Since both the stage of BSD and TVH need the access to the network device, to avoid the conflict, the control unit of TVH is located as the child of the control unit of BSD. According to the control routine defined above, the child control unit will sleep until the process of its parent is over and will only be waken up by its parent.

Figure 11 shows the sample of the details of 2 cycles on the camera node with only 1 CPU. Since there is only 1 CPU, all stages are executed one by one within the cycle. Notice that, by the above control module, the order of the stages changes for each cycle while the TVH stage is executed later than the BSD stage.

In the next section, experiments are conducted to evaluate the performance of the parallel pipeline for both the above default and several customized thread trees

5. Performance Evaluation

In this section, some experiments are conducted to evaluate the performance of the parallel pipeline processing.

5.1 Setup of Experiments

9 to 27 nodes PCs are utilized, and 9 of them are assigned as camera nodes, i.e. the nodes with camera connected. The specification of each PC are same, and is shown in Table 1. All the implementation is conducted on the Linux kernel of version 2.4.18.

5.2 Performance Evaluation of the Parallel Pipeline System with Image File I/O

To evaluate the throughput of the parallel computation without the limit of the capturing facility, experiments where the input silhouettes are loaded from files are conducted. Instead of the stages of IC and ISG, the stage of Load Input Object Silhouette File(LF) is added as the first stage of the pipeline. The default control module for the camera node in following experiments is shown in Figure 12.

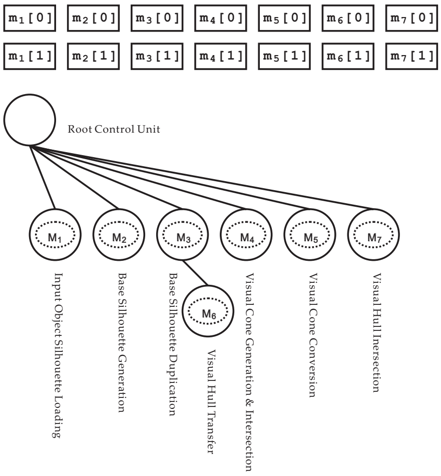

Fig. 12 The Control Module for The Parallel Pipeline with File I/O (Camera Node)

5.2.1 Default Thread Tree for the Parallel Pipeline of 3-Base-Plane Volume Intersection

For each of the 5 multi-viewpoint sequences sets, 3 experiments are conducted. For all the experiments, the voxel size is set as $r = 10$[mm].

5.2.2 Total Throughput

Figure 13 shows the average time of 50 processing cycles. Each cycle starts at the timing of loading one frame on each node which is synchronized and ends at the timing before loading the next frame. The processing time for the 5 postures are shown as bar graphs.

From the graph, the total throughput changes while the posture changes. While the processing time is shortened while adding computation nodes, it time does not decrease from using 18 nodes for each posture. To find the reason of the performance saturation, we pick up the experiments of the posture 1, the details processing time of each stage is shown in the next section.

5.2.3 Processing Time Analysis of the Pipeline Stages

For the detailed processing time inside one processing cycle, the time flow of each stage is measured. Figure 14 shows one snapshot of the processing time of each stage within one cycle on the camera node 1, loading the silhouettes of posture 1 as the input, where total 9 camera nodes and 9

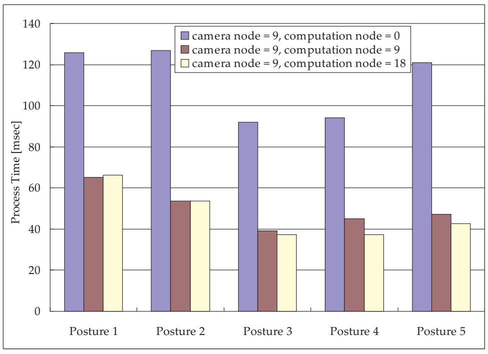

Fig. 13 Total Throughput of Parallel Volume Intersection, where silhouettes of 5 different postures are taken as the input, and the voxel size $r = 10$[mm]

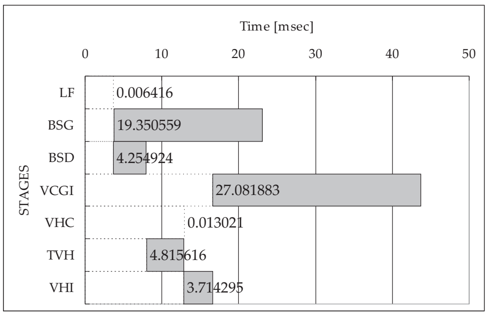

Fig. 14 SnapShot of Processing Time for Each Stage in Parallel Computation of Posture 1, total nodes number = 18

computational nodes are used. In the figure, each flowing bar shows the start and the end timing of each stage. The data label on each bar is the interval of the stage in millisecond.

Since the kernel of the Linux-2.4.18 does not switch the CPU time from the busy thread, which is the thread where no system sleep is called, to other threads, on the system with up to 2 CPUs, up to 2 threads can be executed exactly simultaneously. That is why up to 2 stages are observed being executed simultaneously. From the graph, the BSG and the VCGI stage are much heavier than other stages. Since each node PC has 2 CPUs, the 8 stages are not executed exactly simultaneously. However, from the overlap of the flowing bars in the graph, we can observe that up to 2 stages are executed at exactly same time. Also we can observe that the TVH stage starts after the BSD stage as it is designed, which proves the mechanism to avoid the resource conflict works well.

As mentioned before, by the default structure of the control module, the executing order of the stage within one cycle is not aligned. So within the same cycle, one specific stage, for example the BSD stage, on different node may start at different time. Such time difference will lengthen the remote data I/O stages because extra wait time is taken to synchronize the communication between nodes. To confirm whether such extra wait is caused, the snapshot of the BSD stage on all nodes is shown in Figure 15. It is clear that on each node, the BSD stage does not starts at the same time. And the extra wait time can be confirmed in all 3 experiments.

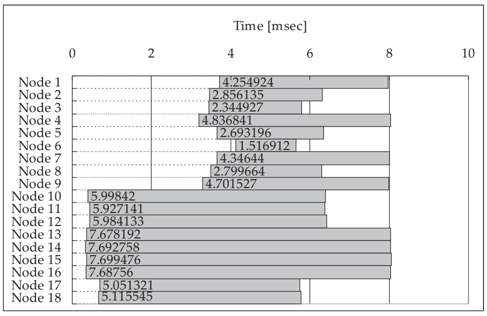

Fig. 15 Processing Time of The BSD Stage, nodes number = 18

Furthermore, since there exist 2 CPUs on each node, up to 2 stages can be executed exactly simultaneously. However, from Figure 14, for much time within one cycle, only one stage is executed. This means a proper executing order of the stages can shorten the processing time and a bad order will damage the throughput.

From the above discussion, the following 2 approaches can be taken to improve the throughput.

(1) Align the start time of the remote data I/O stages, i.e. the BSD and the TVH stages, at the same time among all nodes to get rid of the extra time loss for the communication synchronization.

(2) Tuning the executing order of the stages, so as to keep up to 2 stages executed exactly simultaneously on the 2 CPUs.

In what follows, experiments are shown to realize both of the above approaches by customizing the control module, where only the structure of the thread tree is changed.

5.2.4 Start Timing Alignment for Remote Data I/O Stages

To align the start timing of the BSD and the TVH stages among all nodes in every processing cycle, the control modules for the camera node and the computation node are customized as the Figure

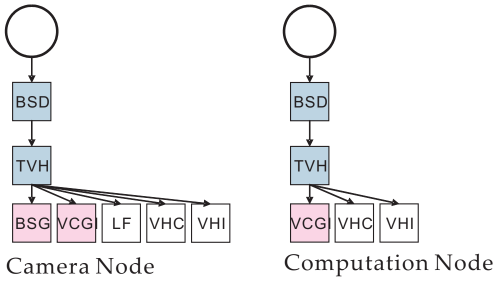

Fig. 16 Customized Control Module for aligning the start time of data sharing stages

16 shows. According to the design of the root control unit, all children of the root are waken up after finishing the process of the root, i.e. conducting the synchronization among all nodes for each processing cycle. By the thread tree shown in Figure 16, where the BSD stage is located as the single child of the root, the BSD stage on each node is executed immediately after the process of the root which synchronizes all nodes. As a result, the start timing of the BSD stage on all nodes are aligned at the same time. And the other communication stage, the TVH stage is located as the single child of the BSD, which is required to avoid the hardware conflict. Since after the BSD stage all nodes are synchronized by the communication task, the start timing of the child of the BSD stage, i.e. the TVH stage, is also aligned at the same time on all nodes.

Applying this customized thread tree on both the camera and the computation nodes, the above experiment is conducted. Figure 17 shows the snapshot of executing time of the BSD stages on all nodes. From the graph, on all nodes, the BSD stage is invoked almost simultaneously and the processing interval of this stage is shortened to almost half of that for the default thread tree, shown in Figure 15. This proves both the efficiency of the start timing alignment of the remote data sharing stage and the effectiveness and flexibility of the tree structured control module, because such start timing alignment is realized by just changing the structure of the thread tree, which is very easy for implementations.

5.2.5 Computational Complexity Based Customization of the Thread Tree

According to the design of the tree structured control module, to keep two stages executed simultaneously, two branches of the stages are needed and the load of each branch must be well balanced.

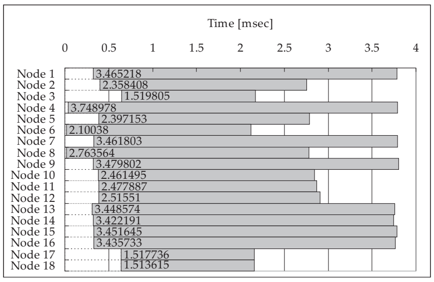

Fig. 17 Processing Time of The BSD Stage, nodes number = 18 with customized control module

From the result of the above experiment, to align the start timing of the communication stages on all nodes, the BSD stage must be the child of the root control unit. So one branch must start from the BSD stage. Meanwhile, for the camera node, the BSG and the VCGI stages are two main arithmetic stages and are much heavier than others. To balance the load of the two branches, it is required to locate each of the two stages in different branch. In practice, since the processing time of each of stages changes while the posture of the object changes. For the same posture, the time of the VCGI on each node may be shortened by increasing the computation nodes. Therefore, the load of the BSG and the VCGI changes case by case, and the location of the BSG and the VCGI stages should also be determined case by case. Let t_{BSG} and t_{VCGI} denote the time of the 2 stages respectively. And let t_{comm} be the summation of time for communication stages of the BSD and TVH. For convenience, the following 4 cases are picked up, (1) $t_{VCGI} > t_{BSG}$ and $t_{VCGI} < t_{BSG} + t_{comm}$, (2) $t_{VCGI} > t_{BSG} + t_{comm}$, (3) $t_{VCGI} < t_{BSG}$ and $t_{BSG} < t_{VCGI} + t_{comm}$, (4) $t_{VCGI} + t_{comm} < t_{BSG}$. 4 types of control modules are designed and shown in Figure 18 for the camera node, corresponding to the 4 cases.

Figure 19 shows the snapshot on the camera node 1 in case of 9 camera nodes and 9 computational nodes are used. In this case, the type 1 control module is selected. From the graph, by applying the customized control module, 2 stages are kept being executed simultaneously for almost whole duration of the cycle. And the processing time of the cycle is shortened from over 40 [msec] to about 30 [msec]. That is, the video frame rate of the 3D shape reconstruction is realized.

Finally, the processing time of one cycle in 3 experiments is shown in Figure 20. In each of the experiment, totally 9, 18 and 27 nodes are used for the

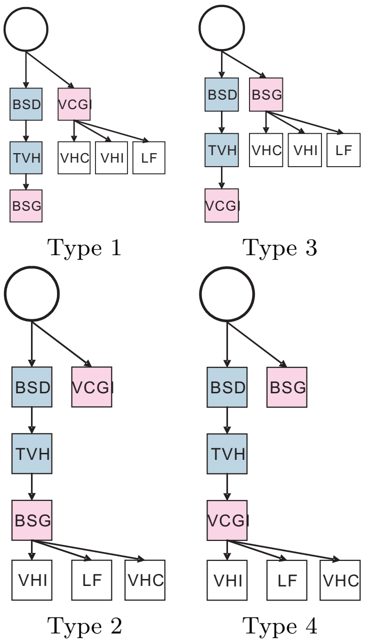

Fig. 18 Customized Control Modules

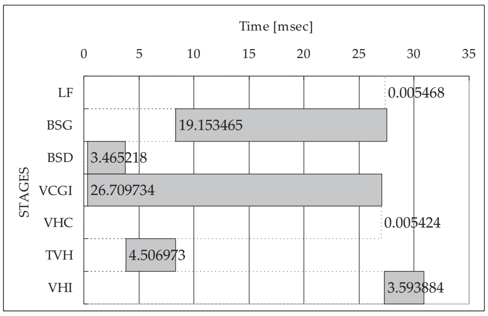

Fig. 19 Snapshot of Processing Time of Each Stage, in case of nodes number = 18, on camera node 1, with the customized control module type 1

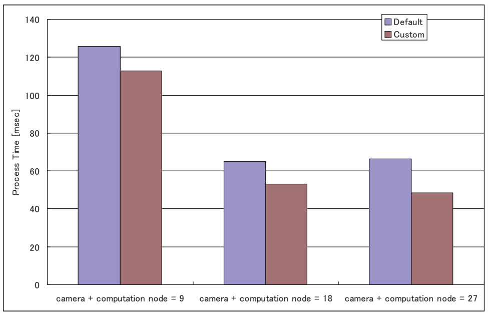

Fig. 20 Processing Time of One Cycle, with customized control module on each camera node

computation. The processing time of one cycle by applying the default control module on each node is also plotted for the comparison. From the graph, we can observe that the performance is improved by applying the customized control module, where on each node, the executing order of the stages is tuned so as to keep 2 stages executed simultaneously.

The above experiments prove that the effectiveness of tuning the executing order of the stages. And the flexibility of the proposed tree structured control module is also proved.

5.3 Performance Evaluation of the Parallel Pipeline System with Online Image Capturing

Based on the evaluation results of the parallel pipeline volume intersection so far, the control module of the camera node is designed for the 3D shape reconstruction with online capturing. As shown in Figure 21, the image capturing (IC) and the input silhouette generation (ISG) stages are added. Since dual CPUs are powered on each PC node, the control module consists of two branches of the stages, like the type 2 control module in Figure 18, the BSG stage is located after the communication stages in one branch, and the VCGI stage is located in the other branch. Notice that, since the main process in IC stage is to load the image data from capturing device to the memory which is usually realized by the DMA (Direct Memory Access) controller in recent PC architecture, during the process on the IC stage, the CPU is free for other processes, therefore the BSG stage is located as the sibling of the IC stage so as to be executed without waiting the image capturing. For the computation node, the type 2 control module in Figure 18 is applied.

As the experiments with the file I/O, 9 to 27 nodes are used in this section. The same 9 camera nodes are used. The camera layout and the bounding box for the shape reconstruction are also as same as the experiments so far.

The performance evaluation is conducted by 18 experiments. For each of them, 9 to 27 nodes, which consists of the 9 camera nodes and additional computation nodes, are used for the parallel 3D shape reconstruction. Each experiment was conducted while capturing a person moving inside the bounding box by the multi-viewpoint cameras system.

At first the voxel size is set as $r = 10$[mm]. The processing time of one processing cycle for each of

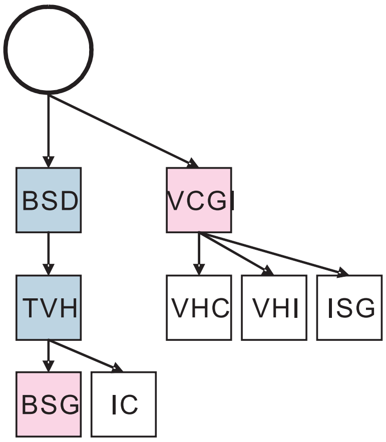

Fig. 21 Cutomized Control Module of Camera Node, for the shape reconstruction with online capturing.

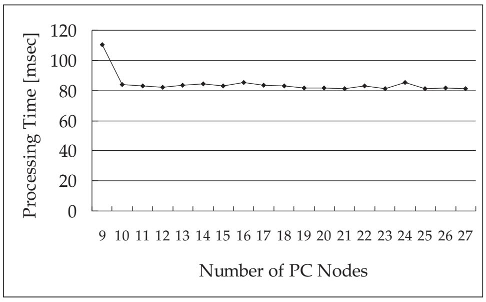

Fig. 22 Processing Time of One Cycle, for the shape reconstruction with online capturing, $r = 10$[mm].

the 18 experiments is shown in Figure 22. The processing time saturates at about 80 [msec] while increasing the number of PC nodes. To find the reason of the saturation, we analyze the processing time for each stage.

Figure 23 shows the snapshot of 2 continuous cycle , i.e. the n-th and the $(n + 1)$-th cycle, on the camera node 1 in the experiment where 12 nodes(9 camera nodes and 3 computation nodes) are used. In the graph, the processing time of each stage is shown as flowing bars as the experiments in the previous section. In addition, the shutter interval is also shown in the graph as the bar with thin sloping stripes. Since the synchronized multi-viewpoint capturing is desired for the parallel 3D shape reconstruction, each processing cycle on each camera node must be synchronized by the shutter timing. As the shutter rate of the camera head in our multi-viewpoint cameras system is fixed to about 12[fps], the shutter interval is fixed at about 80[msec]. In the graph, while all stages ended within the shutter interval, the start of the next cycle is synchronized at the next shutter timing.

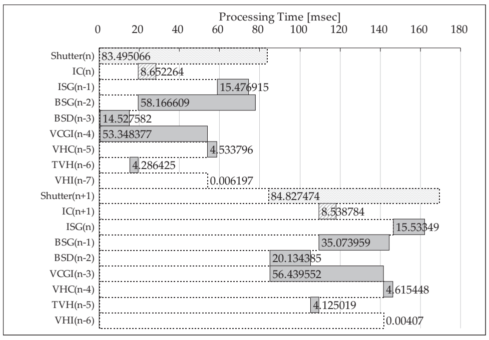

Fig. 23 Snapshot of Processing Time of Each Stage on Camera Node 1, for 2 continuous cycles, i.e. the n-th and the $(n + 1)$-th cycle, $r = 10$[mm]. The "Shutter" bar shows the shutter interval of the camera head.

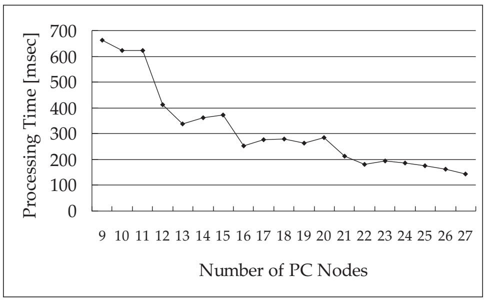

Fig. 24 Processing Time of One Cycle, for the shape reconstruction with online capturing, $r = 5$[mm].

Furthermore, from the graph, the BSG stage was executed simultaneously with the IC stage(shown as the bar with fat sloping stripes), while the VCGI stage was also executed. This shows that the CPU is almost not used for the IC stage for the data I/O from the capturing device is conducted by the DMA controller. This also proves the efficiency of CPU usage in the parallel pipeline processing, where not only the CPU loss time for the remote data sharing is as short as possible but also the CPU loss time for the local data I/O is nearly zero.

Secondarily, the space resolution is set as $r = 5mm$. As the above evaluation experiments, the additional computation nodes are increased from 0 to 18 while 9 camera nodes are used. The processing time of one cycle is shown in Figure 24. From the graph, the processing time is shortened effectively by adding computation nodes.

Figure 25 shows the snapshot of 2 continuous cycle on the camera node 1 in the case of using 27

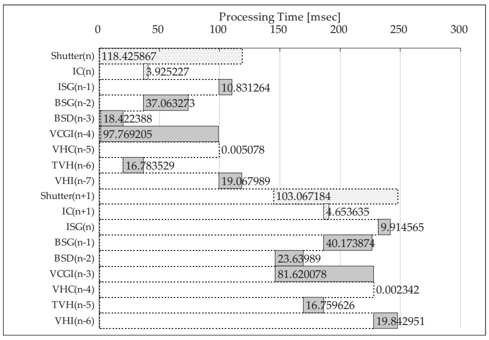

Fig. 25 Snapshot of Processing Time of Each Stage on Camera Node 1, for 2 continuous cycles, $r = 5$[mm]. The "Shutter" bar shows the shutter interval of the camera head.

nodes (9 camera nodes and 18 computation nodes). In the graph, the processing time of each stage is shown as flowing bars as the experiments in the previous section. From the graph, in this case, the throughput is ruled by the time of "VCGI". That is, the performance can be improved by adding more computation nodes.

As the summary, the real-time 3D shape reconstruction while capturing a moving person by the multi-viewpoint cameras system is realized by the proposed parallel pipeline volume intersection on the PC cluster in case of $r = 10[mm]$. The saturation of the performance is caused by the shutter interval of the camera head. In case of $r = 5[mm]$, by adding computation nodes, the throughput is increased effectively.

6. Conclusion

In this paper, the acceleration of the volume intersection is conducted in the following steps.

(1) **Plane-based volume intersection**
The plane-based volume intersection is proposed. By decomposing the 3D space into parallel planes, the 3D shape is represented as cross-sections on the parallel planes. The volume intersection is then realized by computing the intersections of the back projected silhouettes from each viewpoint on the parallel planes. The back projection of the silhouette is conducted as a plane-to-plane perspective projection(PPPP). The PPPP is divided as two kinds of projections, i.e. the projection from the input screen to one of the parallel planes which is called as the base plane, and the projection from the base plane to other parallel planes. The back-projected silhouette on the base plane is called as the base silhouette. The processing to create the base silhouette is then named as the base silhouette generation (BSG), and the processing to create the silhouettes on other parallel planes is named as the visual cone generation (VCG). The linear-wise PPPP and the plane-wise PPPP are proposed to accelerate the BSG and the VCG respectively.
The 3-base-plane volume intersection is also proposed to avoid the breakdown of the PPPP if any input screen of the multi-viewpoint cameras is nearly parallel to the parallel planes. By introducing 3 mutually orthogonal planes as the candidates of the base plane for each camera, not only the breakdown is avoided but also the upper limit of the computation complexity can be estimated.
The efficiency of the accelerated volume intersection is proved by evaluation experiments.

(2) **Parallelization of the plane-based volume intersection by base silhouette duplication**
To realize the real-time volume intersection, the parallel processing on the PC cluster is realized by developing the parallel algorithm for the plane-based volume intersection. For the parallelization, the communications phases, i.e. the base silhouette duplication (BSD) and the transfer visual hull (TVH) are introduced. Since both the base silhouette and the visual hull are represented as two-valued variables, the data size of the communication is small and the additional communication phases cause small overheads, which is proved by the evaluation experiments.

(3) **Parallel pipeline processing model**
To improve the throughput, the multi-phases processing on each node are divided into multiple stages. By introducing the pipeline processing on each node, the stages can be executed concurrently and the throughput can be improved. To realize the pipeline on each node, the tree structured control module is proposed. In the control module, all stages are organized as a tree structure. The children are controlled by their parent and the

siblings are executed concurrently. Therefore, the hardware conflict between stages can be avoid easily, which is proved by the experimental results.

(4) **Synchronization of the communication stages**
While the data size for the communications is small, extra time will spent if the communication stages on nodes are not synchronized, which is observed from the experimental results. By changing the structure of the control module, the communication stages on all nodes are easily synchronized, which is also proved by the experimental results. Furthermore, from the experimental results, such synchronization affects little on other arithmetic stages which are executed concurrently and individually on each node.

(5) **Customizing the control module according to the computation complexity**
Performance evaluations are conducted for the 3D shape reconstruction of the different postures and the results shows that the performance is affected greatly by the posture of the object. That is, the computation complexity changes dynamically for the real-time 3D shape reconstruction of a dynamic object. Experimental results of customizing the control module according the computation complexity prove that the high throughput can only be achieved by changing the control module dynamically for the processing with dynamic computation complexity, such as the real-time 3D shape reconstruction of a dynamic object.

By the above acceleration, the real-time 3D shape reconstruction is realized on the PC cluster. In addition, the effectiveness of the proposed parallel pipeline processing is proved, and the tree structure control module is effective for keeping high throughput while the computational complexity changes dynamically.

References

1) E. Borovikov and L. Davis. A distributed system for real-time volume reconstruction. In *Proc. of International Workshop on Computer Architectures for Machine Perception*, pages 183–189, Padova, Italy, September 2000.

2) G. Cheung and T. Kanade. A real time system for robust 3d voxel reconstruction of human motions. In *Proc. of Computer Vision and Pattern Recognition*, pages 714–720, South Carolina, USA, June 2000.

3) T. Kanade, P. Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, pages 34–47, 1997.

4) Itaru Kitahara, Yuichi Ohta, and Takeo Kanade. 3d video display of sports scene using multiple video cameras. In *Meeting on Image Recognition and Understanding, vol 1*, pages 3–8, 2000.

5) A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

6) Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Proc. of SIGGRAPH 2000*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., July 2000.

7) S. Moezzi, L. Tai, and P. Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, pages 18–26, 1997.

8) Sakamoto N., Yasuhara Y., Kukimoto N., Ebara Y., and Koyamada K. 3d modeling and displaying system for volume communication. In *4th International Symposium on Advanced Fluid Information and Transdisciplinary Fluid Integration*, pages 159–164, 2004.

9) Steven M. Seitz and Charles R. Dyer. Toward image-based scene representation using view morphing. In *Proc. 13th Int. Conf. on Pattern Recognition, Vol. I*, pages 84–89, 1996.

10) S. Sugawara, Y. Suzuki, G.and Nagashima, M. Matsuura, H. Tanigawa, and M. Moiriuchi. Interspace: Networked virtual world for visual communication. pages 1344–1349, December 1994.

11) Sundar Vedula. *Image Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2001.

12) T. Wada, X. Wu, S. Tokai, and T. Matsuyama. Homography based parallel volume intersection: Toward real-time reconstruction using active camera. In *Proc. of International Workshop on Computer Architectures for Machine Perception*, pages 331–339, Padova, Italy, September 2000.